### SSL证书原理及格式

SSL数字证书是在我们上网时用来验证服务器的真实身份的。SSL是传输层的安全协议，后来改进为TLS，不过它用的数字证书我们仍然称为SSL证书。

**SSL协议的简单原理**如下：

客户端发送（自己支持的ssl/tls协议及相关参数）给服务端

服务端（选择其中一个协议版本及相关参数）发给客户端

服务端发送（自己的ssl证书）给客户端

客户端（在本地查找可靠的ca证书去验证服务端的证书是否可信）

若服务端证书可信

客户端发送（用服务端证书里的公钥去加密密钥交换的信息）给服务端

服务端发送（用服务端自己的私钥加密的密钥交换信息）给客户端

双方计算交换后的密钥（为对称的加密算法）

客户端用交换后的密钥发一些验证信息给服务端，服务端正确响应，则SSL验证通过。

接下来：

双方 使用对称的加密算法去加密正式的数据

当ssl会话到期后，再重新交换新的密钥

ssl用到了非对称的密钥体制中的RSA算法，以及DH密钥交换算法（当然，这个密钥交换可能升级了，不再是原来的DH算法了，可能是ECDHE），双方交换的是什么密钥？是对称加密算法的密钥，具体是哪个对称加密算法是由ssl握手时协商的，可能是DES，也可能是AES。

**RSA算法：**

利用大整数因子分解的困难性

先随机生成2个大素数p和q，再计算n=p\*q

ψ(n)=(p-1)\*(q-1)

随机数e (0<e<ψ(n))满足gcd(e,ψ(n))=1，即e与ψ(n)互素

计算d=e-1 mod ψ(n)

然后就得到 公钥pub-key为 e,n

私钥pri-key为d,n

加密算法：密文=明文e mod n

解密算法：明文=密文d mod n

加密和解密是相对的，互逆的，当用e去加密时，就得用d去解密，当用d去加密时，就得用e去解密。要想通过e得知d是不可能的，要想通过d得知e也是不可能的，所以服务端生成了密钥对ed(n)后，把其中的公钥给客户端，自己保留私钥，这样就可以互相加密解密了，

服务端用私钥加密的信息，所有的客户端都可以看到，因为客户端很轻松就能获得服务端的证书里的公钥，但客户端发的加密消息（用公钥加密），只有服务端能看到，因为私钥只存在服务端手里。

所以RSA算法不是用来加密消息的，因为客户端都能看到服务端的消息，它只是用来做数字签名的，顺便交换密钥。

、

**DH密钥交换算法**：

利用离散对数问题的难解性

q为一个素数，a为q的一个本原元，要交换密钥的双方 已知a,q

Y = ax mod q

给出x容易算出Y，给出得知Y时，反而不易（很难）算出X

密钥交换过程如下：

用户A和用户B协商q和a

用户A产生一个随机数X(A)<q 用户B产生一个随机数X(B)<q

用户A计算Y(A)=ax(A) mod q 再把Y(A)发给用户B

用户B计算Y(B)=ax(B) mod q 再把Y(B)发给用户A

用户A根据B发来的Y(B)计算出 k=(Y(B))X(A) mod q

用户B根据A发来的Y(A)计算出 k=(Y(A))X(B) mod q

然后，用户A和用户B算出的k是一样的，这个k就是交换后的密钥

当其他人截获了AB的信息，得知了q,a,Y(A),Y(B)也无法计算出k

因为其他人不知道X(A)和X(B)，前面讲过，给出Y，是不易算出X的。

密钥交换过程中就算使用明文，其他人也无法得知AB双方交换后的k，比较安全，但单纯使用密钥交换算法是无法应对中间人攻击的，所以得用上RSA算法，其中服务端使用私钥去加密要交换的Y(A)，客户端用公钥去加密要交换的Y(B)，这样就可以应对中间人攻击，接下来要信任的就是服务端了，因为其他人也可以宣称自己就是这个服务端，它做了流量的劫持，不再做中间人了，自己替代了目标服务器。所以我们需要一个共同信任的机构，由这个机构去对所有的其他服务器的公钥及相关信息做一个签名，生成一个证书，没错，就是ssl证书。

（这个交换的密钥是什么？ 当然是对称的加密算法的密钥了，ssl握手完成后，通信双方就使用对称的加密算法去加密 正式的数据了，因为对称加密算法执行速度比非对称的快）

**签名**：

可信的CA（验证机构）使用它的私钥把 其他服务端的（公钥及相关信息）加密（这就叫签名），生成一个ssl证书，服务端收到CA发的证书后，部署到服务器上，当有客户请求ssl连接时，服务端再把这个证书给客户端，客户端用可信CA的公钥去解密此证书，解密后的内容再和此证书上所宣称的信息一比对，就可得知此证书的真伪了。

可信CA的公钥存放在它的CA证书里，顶级CA证书是由它自己来签名的（中间CA的证书是由上层CA来签名的）。可信的CA一般在操作系统安装时就放在操作系统里了。

话说我们凭什么就要相信这些CA呢？因为这个验证机制是人家最早提出并得到了广泛使用的，不用它，还能用谁呢？

**ssl证书里都有哪些内容**呢？

ssl证书文件里有2部分内容，一部分是明文的，含服务端的相关身份信息及RSA的公钥还有表明由哪个CA给它签名的，以及密文的（"服务端的相关身份信息及RSA公钥"的hash值）这个密文是由（明文里表明的）可信CA去加密的。

这样我们就可以从操作系统里找到此证书所说的那个CA的公钥，用这个CA的公钥去解密此证书的密文部分，得出密文里的hash信息再和 根据此证书的明文部分计算出的hash比对，如果吻合则说明此证书可信。这样服务端的身份就可信了吗？其实没那么快就相信服务端的身份，因为真实服务端的证书我们所有客户端都可以获得的，所以也有可能是其他人用此证书去充当服务端，不过没关系，它没有服务端的私钥，所以我们在ssl验证时，在交换完密钥后，会发一些验证消息，以验证服务端的真实性。因为如果伪造的服务端没有真实服务端的私钥，它发过来的交换信息在客户端这边用真实服务端的公钥解密时，就不能正确解密，所以完成密钥交换后，双方得到的key是不一致的。这样完成密钥交换后，再发一些验证消息，这个伪造的服务端就被识破了。

当客户端找不到证书里表明的CA时，就会提示连接是不安全的，无法验证此网站的身份。

**证书链**：

可信的CA只是少数的，后来申请ssl证书的人越来越多，于是这些顶级的CA又把签名权下放到其他的商家那里，由其他商家充当中间CA，我们现在申请的ssl证书一般都是中间CA签名的，中间CA的证书是由顶层CA签名的。而我们操作系统里可能没有预装中间CA的证书，导致我们无法验证由中间CA签名的ssl证书，这时就需要网站服务端提供证书链，从证书链中获取中间CA的证书，再从中间CA证书去找更上层直至顶层CA的证书。这样就完成了最下层ssl证书的验证。

有时候浏览器会自作聪明地 自己去找中间CA的证书再去找顶级CA，就是说它不看网站服务端给的证书链了，它自己去找合适的证书链。这也就是 有时我们网站上的证书链明明是4层的，结果在浏览器里只有3层 的原因了。（火狐浏览器可以导出它自己寻到的证书链）

**ssl证书的格式**

ssl数字证书文件的内容可以是文本格式，也可以是二进制格式。

常用的证书标准及文件格式如下：

|  |  |  |
| --- | --- | --- |
| 编码格式 | 证书文件后缀 | 说明 |
| x.509 | .cer .crt .der .pem | 证书里含RSA的公钥，相关身份信息 |
| pkcs#7 | .p7b | 含RSA的公钥，可含CRL信息 |
| pkcs#12 | .p12 .pfx | 可含RSA的私钥，有密码保护 |
| jks | .jks | java特有的，可含私钥 |

怎么申请自己网站的ssl证书呢，

1.先在本地生成.key密钥对文件，这个xx.key文件叫作密钥对文件（有些人称它为私钥

文件是不正确的），因为这个.key文件是RSA算法生成的公钥及私钥文件，公钥和 私钥必需是成对生成的，不可能先生成一个私钥，再随后生成公钥。

2.把xx.key密钥对文件里的公钥取出来，加上一些网站的身份信息，合成一个.csr文

件，.csr文件叫作证书请求文件，里面的中文信息用utf-8编码。

3.我们把.csr文件提交给某个可信CA，由此CA去签名，生成一个证书文件，CA服务商

一般会提供多种格式的证书文件以及它们的证书链文件。

4.我们再根据自己web服务器的要求去转换成相应的格式，比如tomcat的.jks格式和IIS

的.pfx格式。

pkcs#7证书可以为二进制文件，也可为base64的文本文件

pkcs#12的证书不论后缀为何，都是二进制文件，

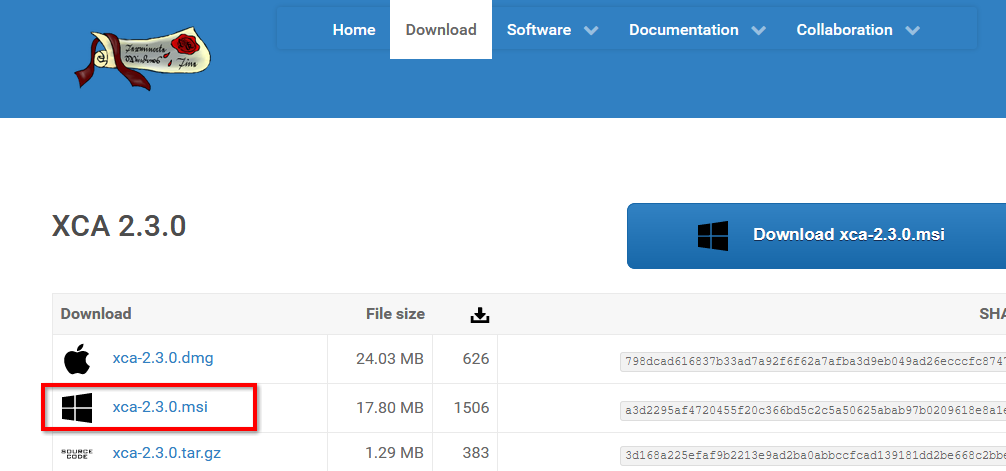
x.509编码的话，.cer和.crt一般都是使用PEM（base64）编码的文本文件，.der为二进制文件

.pem后缀证书可以是二进制的，也可以是base64文本的，具体的得打开看看才知道，还有证书链也有以.pem为后缀的。证书链文件就是由多个相关证书的内容放在一个文件里生成的。

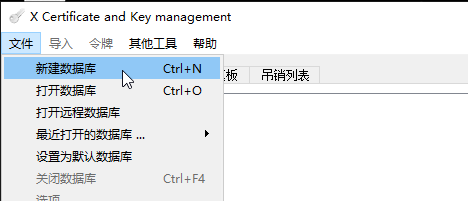
ssl证书格式怎么相互转换呢，可以使用XCA证书管理工具，也可以使用openssl工具，这两者都是开源免费的。

**XCA证书管理工具**的使用

到官网<https://hohnstaedt.de/xca/index.php/download> 去下载新版的xca工具并安装



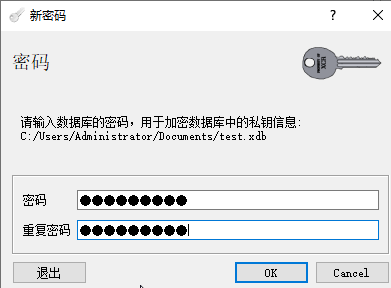
下载xca-x.x.x.msi这个安装包，安装后，打开xca软件



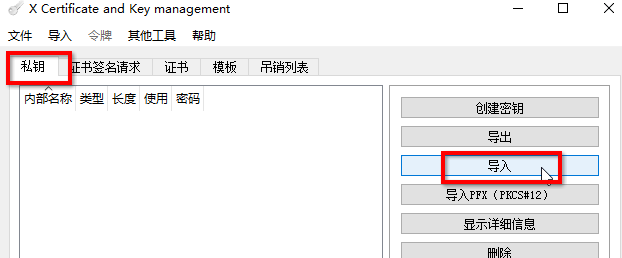
在主界面上，点击左上角菜单栏的“文件”→“新建数据库”

选择保存到某个目录下，名为test.xdb

确定后，要求输入数据库的密码

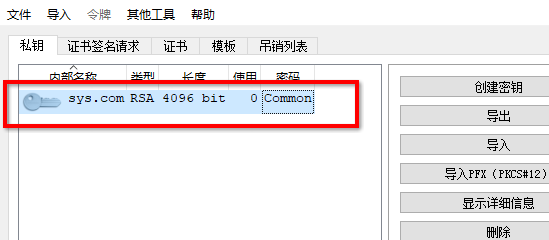


点击OK后，就行了，接下来导入我们从CA证书商家那获得的ssl证书，



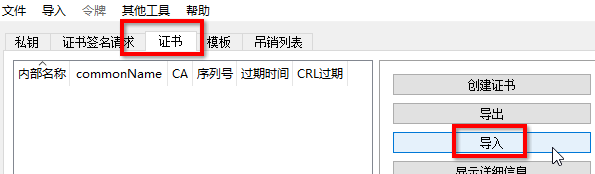
点击“私钥”，“导入”，选择目标.key/.pem/.der/.p8/.pvk/.pub文件，例如sys.com.key

其实导入的不止是私钥，因为.key文件是密钥对文件，在xca里只显示其中的私钥，和私钥配对的公钥也是包含在此文件里的。



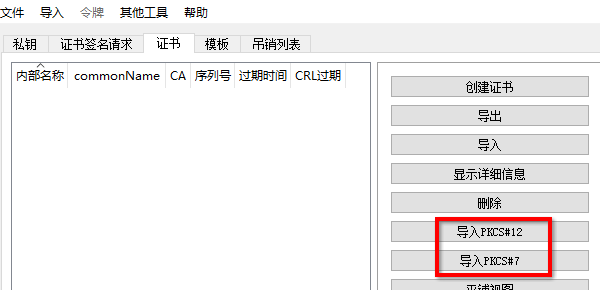
导入成功后，就会出现一个key列表信息，如上图

再导入证书文件，点击“证书”，“导入”



默认的导入只能导入x.509格式的.pem/.cer/.crt/.der

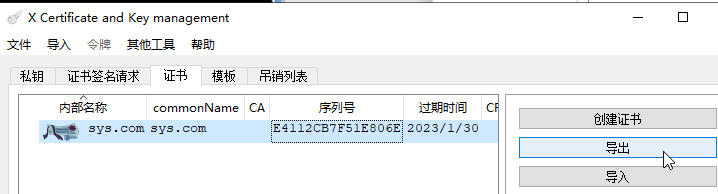
也可以点击下面的“导入pkcs#12”（.pfx/.p12）和“导入pkcs#7”（.p7b）



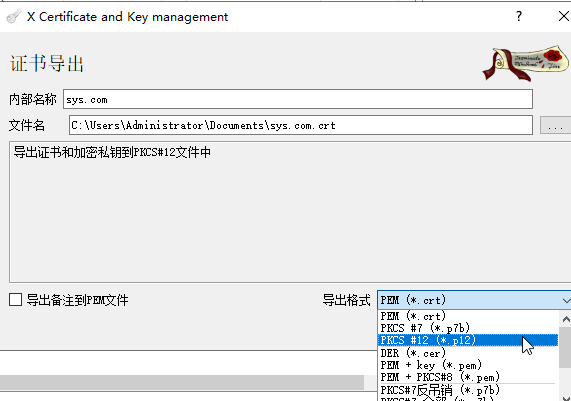
导入成功后，也会多出一个证书文件列表，如下图



在xca管理工具里，导入证书及密钥后，我们可以查看证书或密钥的详细信息，也可导出为其他格式的文件，比如导出为.p12文件，（现在的.p12文件和.pfx文件是一样的，可以直接把.p12后缀改为.pfx，就能在iis上使用了）

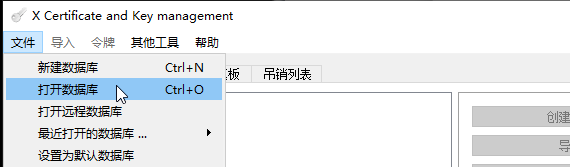


选中目标证书，点击“导出”

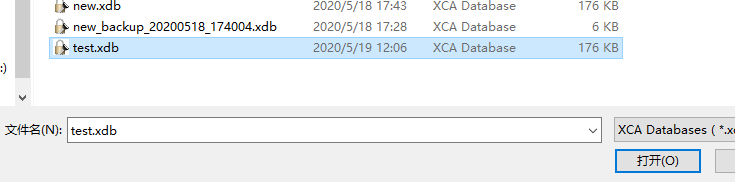


然后选择导出后的目录，以及导出的格式，如pkcs12的.p12后缀格式，它可以直接改名为.pfx

XCA除了可以转换证书格式，也可以生成RSA密钥对，并创建证书请求文件



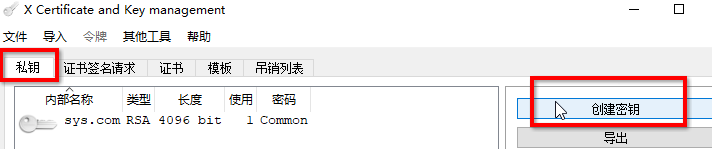
xca在打开时，默认不会打开之前创建的数据库，要我们手动去打开，点击“文件”，“打开数据库”



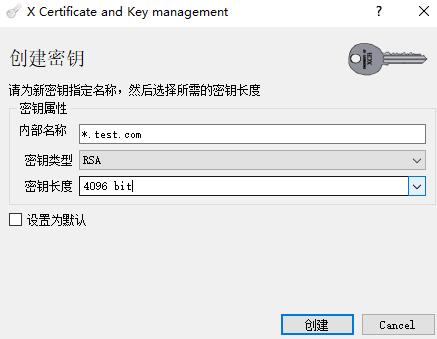
找到目标数据库，点击“打开”



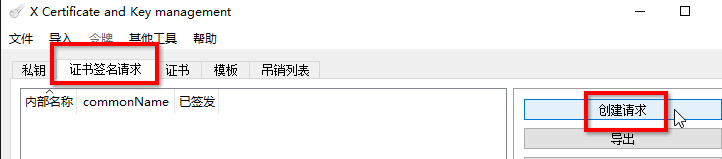
输入数据库的密码，点击“OK”



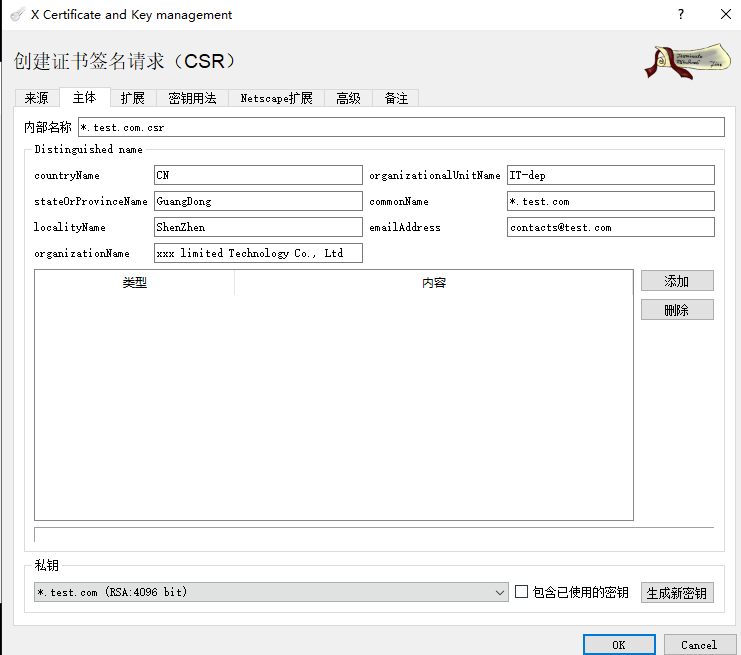
在“私钥”界面，点击“创建密钥”

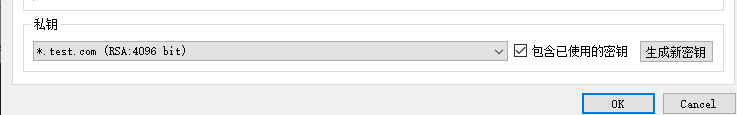


名称随便写，比如\*.test.com，一般是写网站域名或带\*通配符的域名，密钥长度不低于2048位，现在推荐用4096位



然后在“证书签名请求”页，点击“创建请求”





如上图，证书签名请求文件里，要填写的不多，一般只写主体部分，其他的会写的就写，不会的就不写。内部名称就是请求文件在xca数据库里的文件名称，可以随便写，重要的是下面的7个空以及最下行的“私钥”，（这里得再说一下，key文件不叫私钥文件，就叫密钥或密钥对文件，含RSA的公钥及私钥）中文是用utf-8编码

countryName填国家代码，只能写2个英文字母，中国写CN

stateOrProvicnceName填写省级行政单位名，如GuangDong，支持中文

localityName填写地级行政单位名，如ShenZhen，支持中文

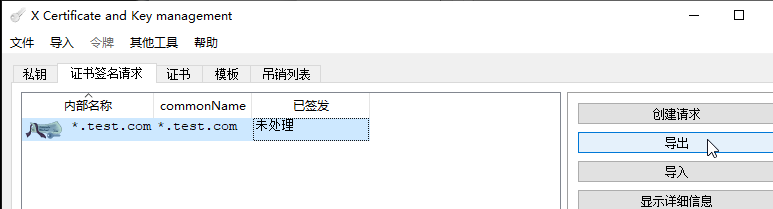
organizationName填写组织名称，一般写公司名，如xxx xxx co., ltd支持中文

organizationlUnitName组织单位名，一般写公司的某个部门，如IT-dep，支持中文

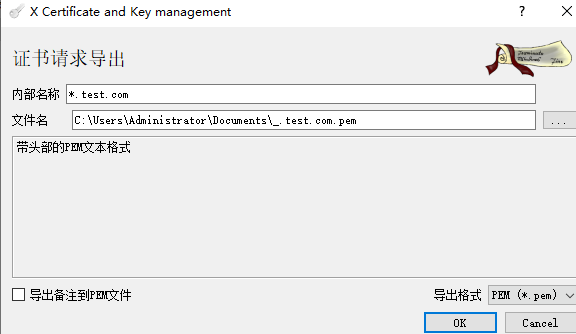
commonName通用名称，一般写域名，可写带\*.的通配符域名

emailAddress可写可不写

最下行的“私钥”是指密钥文件，表示xca管理界面里的“私钥”一栏里的密钥，在证书签名申请文件里，只会导入密钥中的公钥



生成证书签名请求文件后，点击目标文件，点击“导出”

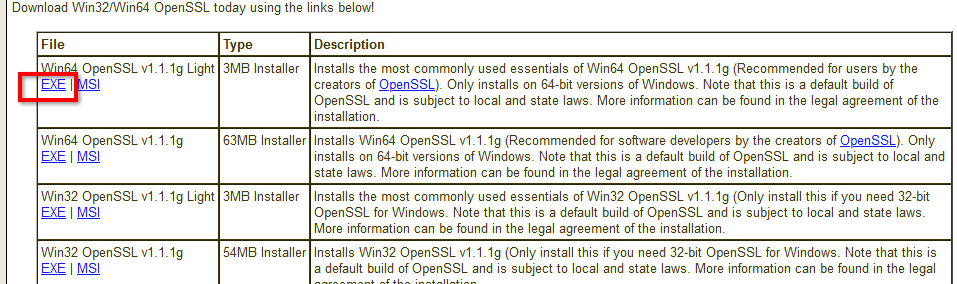


导出csr请求文件后（它的后缀不一定非得是.csr，也可以是.pem）就可以提交给ssl证书签发机构了。证书签发机构审核通过后，就会发给我们证书文件

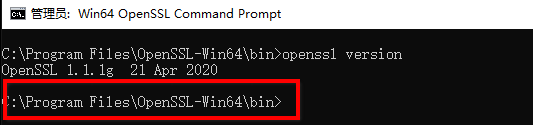
**OpenSSL工具**的使用

Linux系统可以用apt-get install openssl或yum install openssl安装，windows可以到

[https://slproweb.com/products/Win32OpenSSL.html 网址下载，选择](https://slproweb.com/products/Win32OpenSSL.html网址下载，选择)下载exe文件



windows版安装后，在cmd里进入安装目录，再运行openssl.exe命令

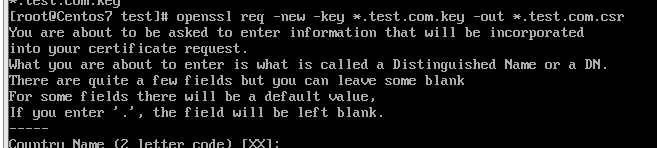


# openssl genrsa -out \*.test.com.key 生成.key密钥对文件，默认是2048位

#openssl genrsa -out \*.test.com.key 4096 指定RSA密钥长度

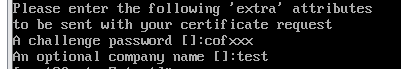


#openssl req -new -key \*.test.com.key -out \*.test.com.csr 创建证书签名申请文件



然后根据提示输入相关信息，





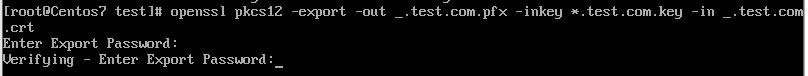
还要输入额外的信息，随便写，也可不写

然后把\*.test.com.csr请求文件提交给CA去签名，通过后得到\_.test.com.crt或.cer之类的证书文件。

格式的转换：

把x.509编码转成pkcs12

openssl pkcs12 -export -out \_.test.com.pfx -inkey \*.test.com.key -in \_.test.com.crt



.pfx文件要求输入保护密码

把.p7b转为x.509的.crt

先转成PEM文本的文件，再转输出为certs格式的

# openssl pkcs7 -inform der -in \_.test.com.p7b -out \_.test.com.pem.p7b

# openssl pkcs7 -print\_certs -in \_test.com.pem.p7b -out \_.test.com.crt

其他的不多说了，我也不会了，哈，自行网上搜索吧。

作者：Cof-Lee

2020年5月19日